Mitglied der Helmholtz-Gemeinschaft

Exa Scale Innovation Center

JÜLICH FORSCHUNGSZENTRUM

System level evaluation of an in-memory-processing architecture

D. Pleiter | ScalPerf15 | 24 September 2015

Outline

- Processing-in-memory (PIM) architectures
- Methodology of evaluation for selected scientific computing applications
- Selected results and architecture assessment
- Summary and conclusions

Credits

IBM

- Jaime Moreno, Rajiv Nair, José Brunheroto
  + others from AMC team
- Hans Boettiger, Thilo Maurer

JSC

- **Paul Baumeister**, **Thorsten Hater**, Andrea Nobile

Application developers

- Giannis Koutsou, Stefan Krieg, Hubert Simma
- Fabio Schifano, Lele Trippiccione
- Stefan Blügel

Most results published at ISC'15:
doi:10.1007/978-3-319-20119-1_8

Processing in memory (PIM)

Architectural arguments in favour of PIM

- Reduce (off-chip) data transport → less energy
- Larger B_{fp} / B_{mem} by addressing Rent's rule
  → higher performance
  - More wires available to connect compute and storage

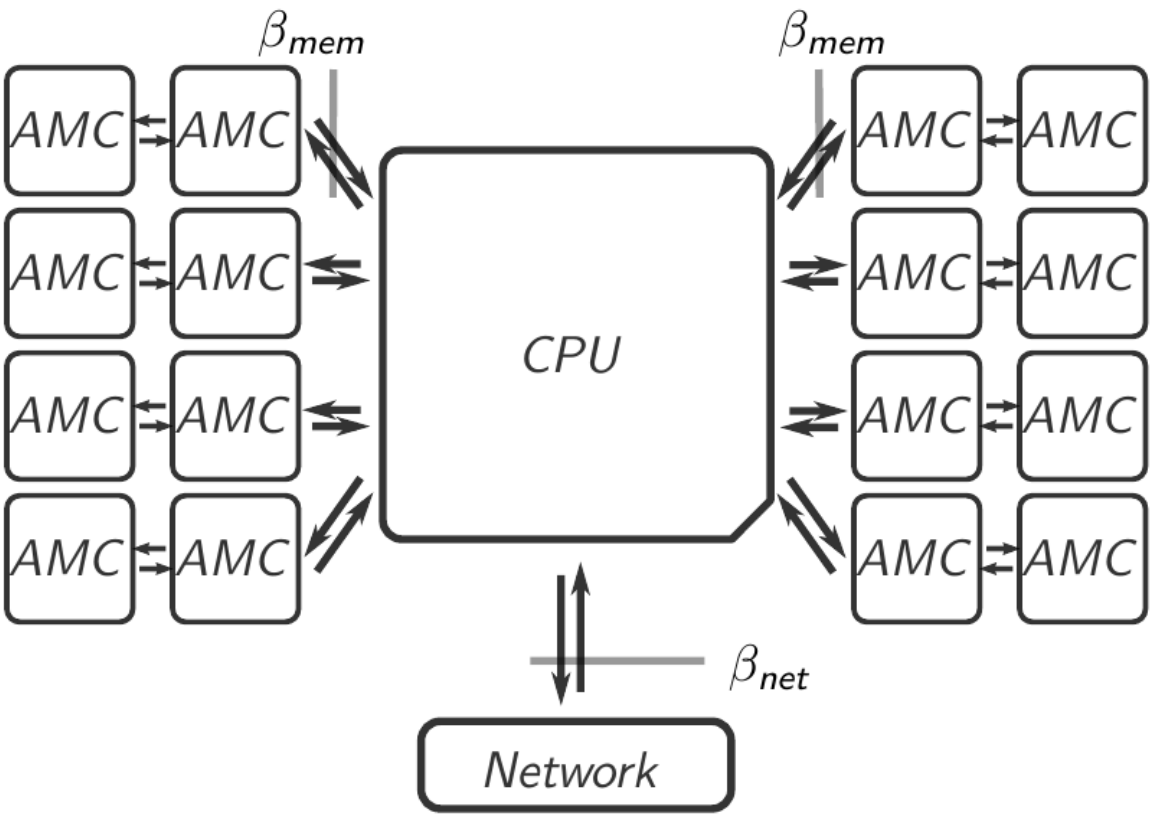
Various projects since 90s

- Computational RAM (1992)
- Intelligent RAM (1997)
- DIVA (1999)
- FlexRAM (1999)
- Gilgamesh (2002)

No products, yet

- Major challenge: high costs, programming model
- New opportunities due to 3d stacking technologies

Node strawman



Components

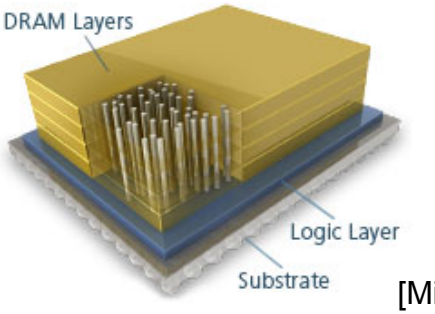
- General-purpose processor
- Network interface
- Memory modules with integrated (simple) processing units

IBM's Active Memory Cube (AMC)

[R. Nair et al., 2015]

PIM architecture based on Hybrid Memory Cube

- HMC = stack of logic die + multiple memory dies
- Communication through through-silicon vias (TSV)



[Micron]

AMC adds 32 compute lanes

- Very Large Instruction Word (VLIW) architecture
- Temporal SIMD concept

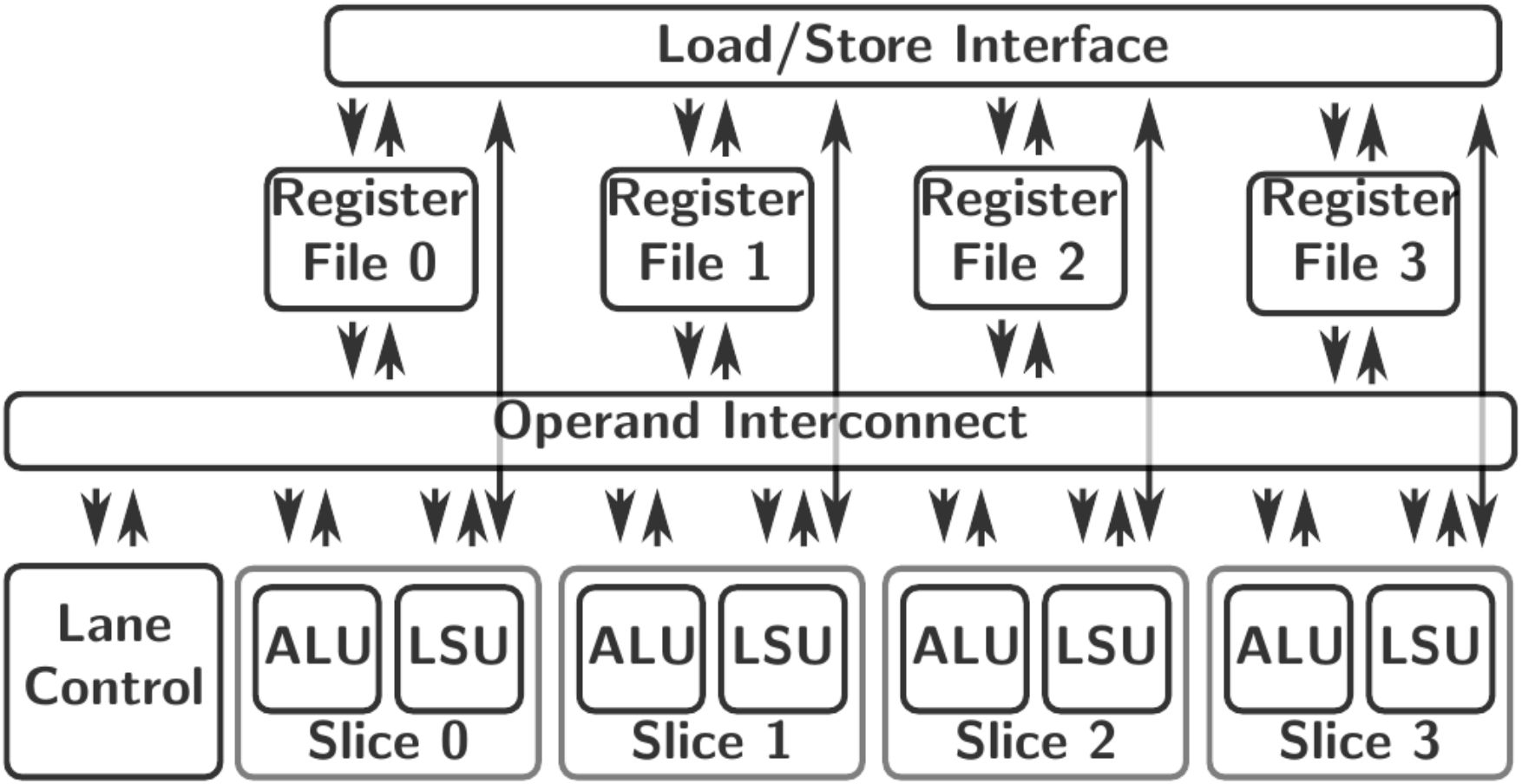
Dual-ported memory concept

- Access from CPU and AMC lanes
- Coherent memory access within same address space

Chainable for capacity and compute performance

Compute lane architecture

```
[ic] br {alu; lsu} {alu; lsu} {alu; lsu} {alu; lsu}
```



AMC performance figures

Floating-point operation throughput

- 4 double-precision FMA per lane and cycle
- 8 single-precision FMA per lane and cycle

Temporal SIMD concept

- Instructions up to 32× repeatable
- Vector registers of length 32

Memory performance

- 8 Byte per lane and cycle → B_{fp} / B_{mem} = 1 (DP)
- Minimal access latency: 24 cycles

Target clock: 1.25 GHz

- Nominal floating-point throughput: 320 GFlop/s (DP), 640 GFlop/s (SP)
- Nominal memory bandwidth: 320 GByte/s

Target power envelope: ~10 W

Approach for application evaluation

Application selection

- Application with needs of increasingly scalable compute resources

Application roadmap assessment

- Interview of application experts based on questionnaire

Application kernel performance evaluation

- Port to AMC and cycle accurate simulation

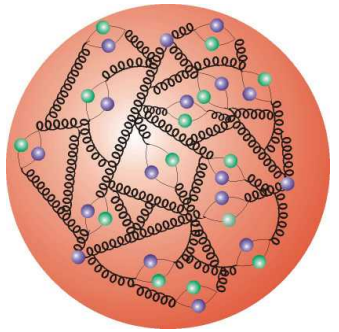
System level performance assessment

- Performance modelling approach

Applications overview

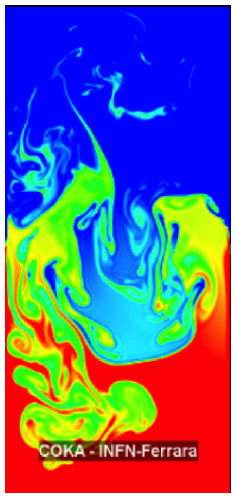
Theoretical high-energy physics

- Simulation of Quantum Chromodynamics: Lattice QCD
- Expected developments towards 2017:
  - Increase of lattice volume up to $96^3 \times 256$ or $128^3 \times 256$
  - Simulations at physical quark masses
  - Trend towards more complex algorithms
- PRACE: >400 (PFlop/s) * year in 2020



Computational fluid dynamics

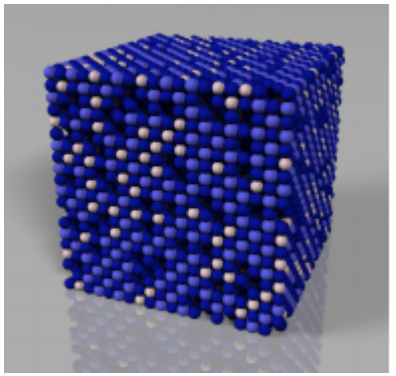
- Current 2d Lattice Boltzmann formulation: D2Q37
- For future 3-dimensional formulation expect need of 20 PFlop/s (DP, sustained) for about 64 days
  → Need for O(100) PFlop/s systems



Applications overview (cont.)

Condensed matter physics / material research

- Density Functional Method based approaches
  - Focus on real-space and KKR formulation
- Use cases
  - Complete simulations of entire nanostructures
    → need for scalability
  - Multiple simulations for several systems and many different sets of parameters
    → need for high throughput
- No relevant limits in intrinsic parallelism
  - Compared to parallelism of system



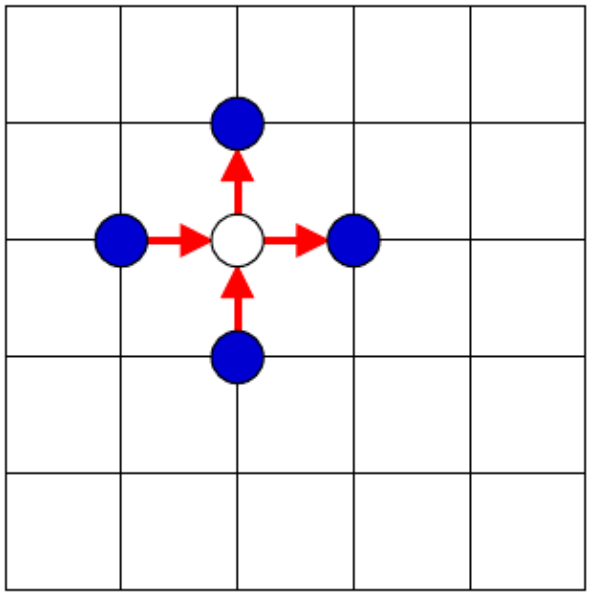
LQCD kernel on AMC

Focus on solver for Wilson-Dirac → SpMV

$$\psi_x = \sum_{\mu=0}^{3} \left\{ (1+\gamma_\mu) U_{x,\mu} \phi_{x+\hat{\mu}} + (1-\gamma_\mu) U^{+}_{x-\hat{\mu},\mu} \phi_{x-\hat{\mu}} \right\}$$
$$= D[U]_{xy} \phi_y$$

Performance signatures

- Balance parameters (SP)
  - I_{fp} = 1320 Flop / site
  - I_{mem} = 1.4 kiByte / site
- Regular control flow
- Complex arithmetics



LQCD kernel on AMC

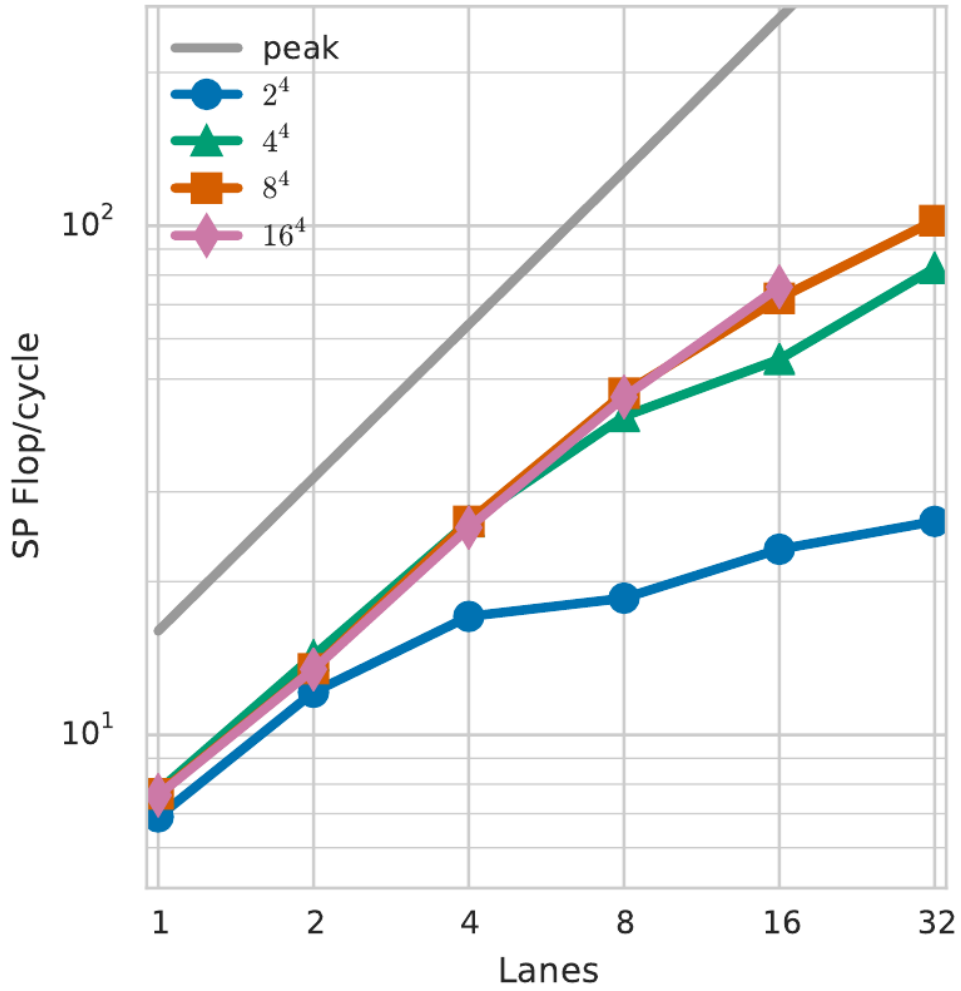
Implementation limited to parts of matrix-vector multiplication

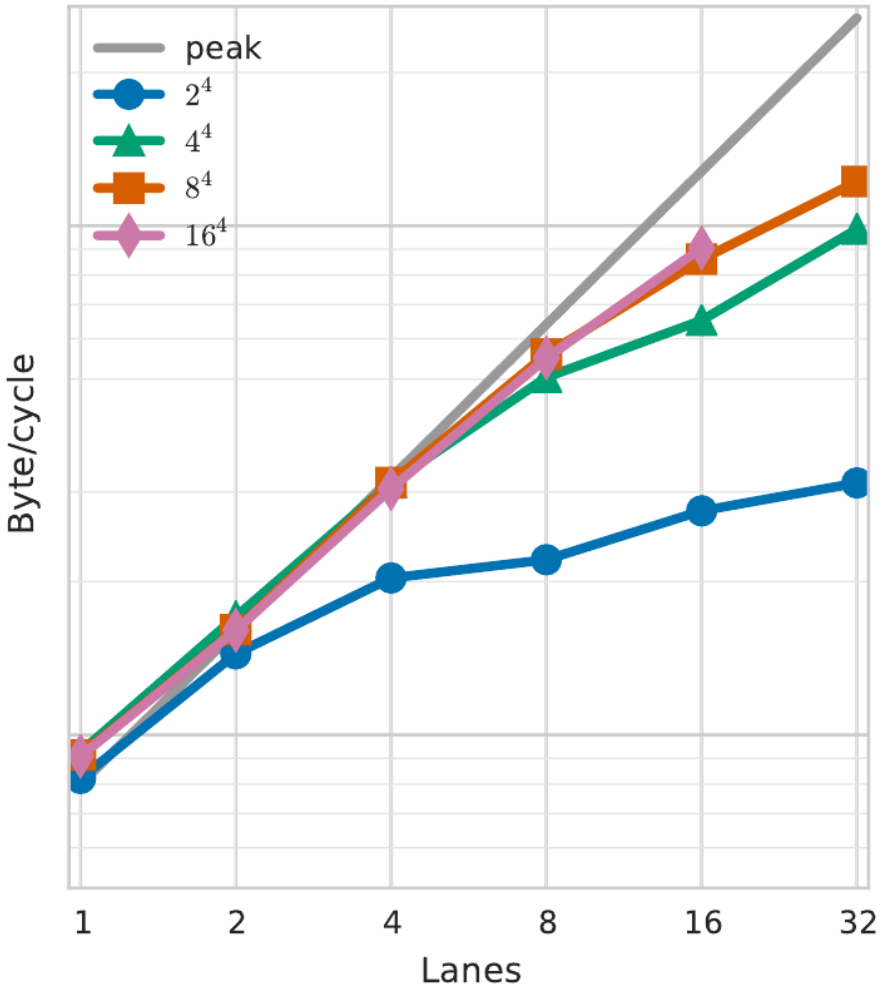
- Projection of spinors and multiplication with *U*
- 4 space-time directions mapped to 4 slices

Complex arithmetics

- For double precision no special support needed
- For single precision additional (spatial SIMD) instructions introduced

LQCD kernel on AMC (cont.)





D2Q37 kernels on AMC

General LBM formulation

$$f_\alpha(\vec{x}_i + \vec{e}_{\alpha,i}\,\Delta t, t + \Delta t) = f_\alpha(\vec{x}_i, t) - \frac{\Delta t}{\tau}\left[f_\alpha(\vec{x}_i, t) - f_\alpha^{(\mathrm{eq})}\left(\rho(\vec{x}_i, t), \vec{u}_i(\vec{x}_i, t)\right)\right]$$

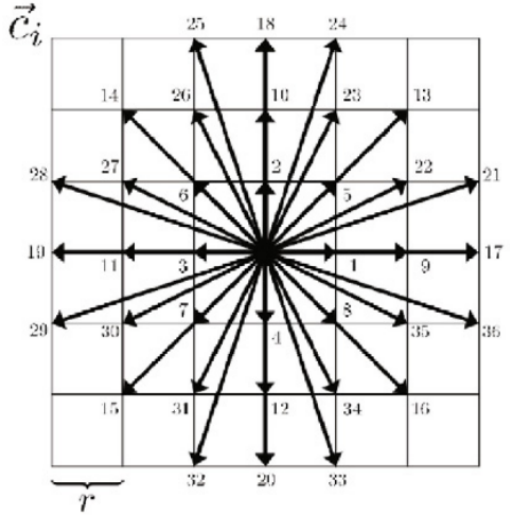
Performance signatures

- Propagate kernel
  - Streaming memory access
  - No arithmetic operations
- Collision kernel in case of D2Q37
  - Balance parameters (DP)
    - I_{fp} = ~6,000 Flop / site
    - I_{mem} = 592 Byte / site

D2Q37 kernels on AMC: Propagate

Computational task:

- Move the 37 populations according to their velocity




```
// Update row counter, setup column counter
R: [ 1]      { sub row, row, 0x1;                      } { mov CTR, col; } {;} {;}
// Inner loop: Read one site and scatter with stencil
C: [32]      {                  ; ldu tmpA, src, 0x8   } {             ; } {;} {;}
   [ 5]      {                  ; ldu tmpB, <src, 0x8  } {             ; } {;} {;}
   [32]      {                  ; st tmpA, velA, tgt   } {             ; } {;} {;}
   [ 5]      {                  ; st tmpB, velB, tgt   } {             ; } {;} {;}
// Update target address and move to next column
[ 1] bdcnz C { add tgt, tgt, pop;                      } {             ; } {;} {;}
// Are we done with all rows?
[ 1]         { cmp row, 0x0     ;                      } {             ; } {;} {;}
// At the end of a row, skip boundary cells
[ 1]         { add src, src, bnd;                      } {             ; } {;} {;}
[ 1]         { add tgt, tgt, bnd;                      } {             ; } {;} {;}
// Wait for cmp and then move to next row
[ 6] bc R    {                  ;                    }  {             ; } {;} {;}
```

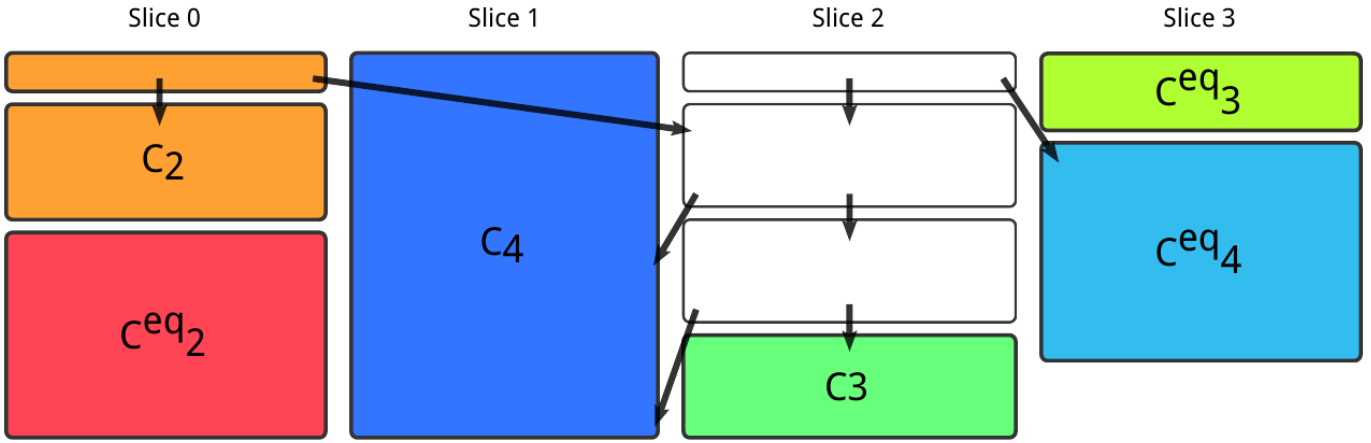
D2Q37 kernels on AMC: Collide

Computational task:

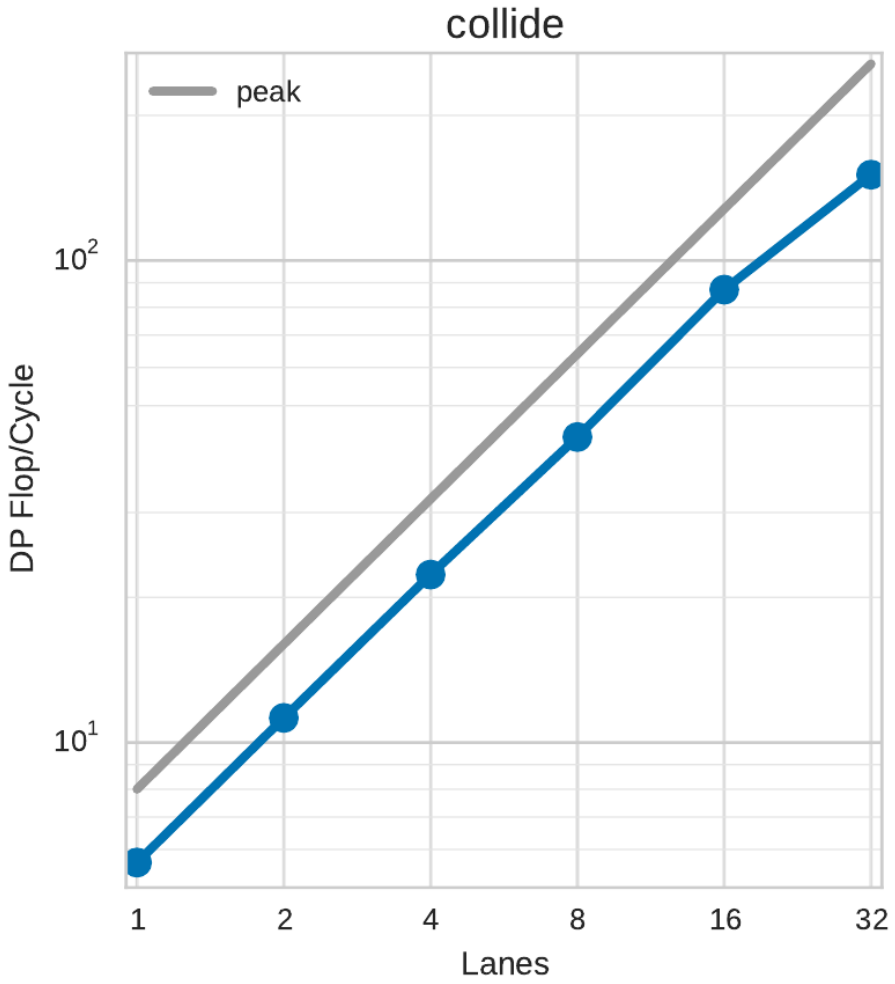
- Compute equilibrium and update local distribution
- D2Q37 compute intensive due to equilibrium distribution being expressed in terms of Hermite polynomials

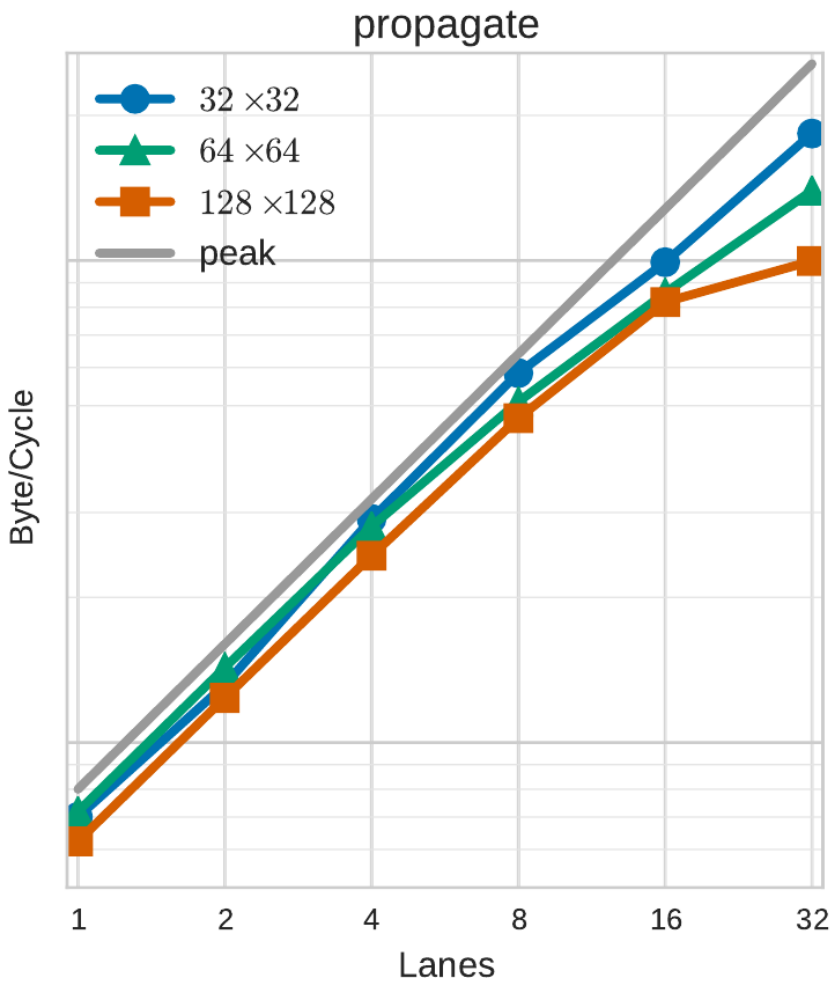
Mapping to AMC

- Dense packing of instructions can be achieved:
  - 10% ALU NOPS
  - 97% LS NOPS

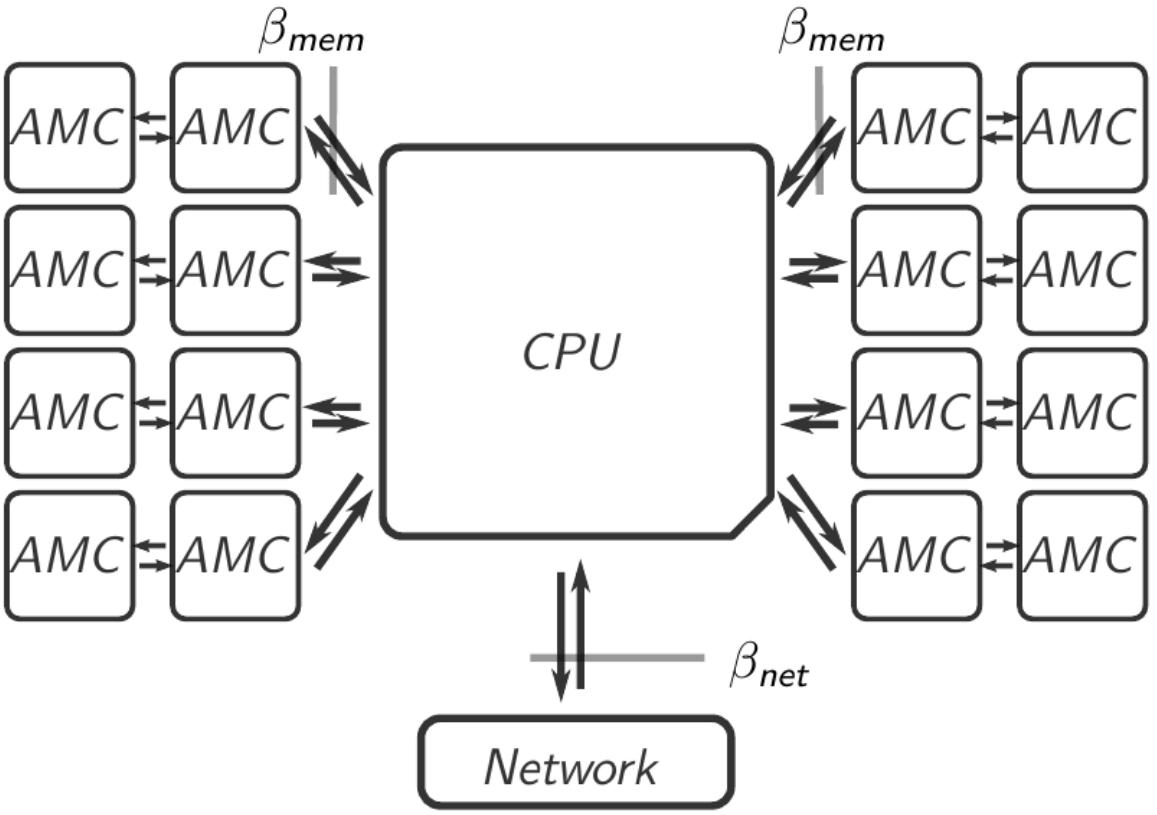


D2Q37 kernels on AMC





System level performance model



Parametric node architecture

- AMC-CPU and network bandwidth
- Number of AMC channels and AMCs per channel

System level performance model (cont.)

Model assumptions

- Latency-bandwidth model ansatz: $T_x = \lambda_x + I / \beta_x$
- Perfect overlap of computation and communication

Ansatz

- T_{comp}: Time needed for computations
  $\rightarrow$ determination based on cycle accurate simulations
- T_{mem}: Time needed for intra-node communication
- T_{net}: Time needed for inter-node communication
- $T = \max(T_{comp}, \max(T_{mem}, T_{net}))$

Parameter choices

- 8 memory channels, 2 AMC per channel
- CPU-network: $\lambda_{net} = 1$ µs, $\beta_{net} = 100$ Gbyte/s
- AMC-CPU: $\lambda_{mem} = 1$ µs, $\beta_{mem} = 32$ Gbyte/s

System level performance: LQCD

Observations

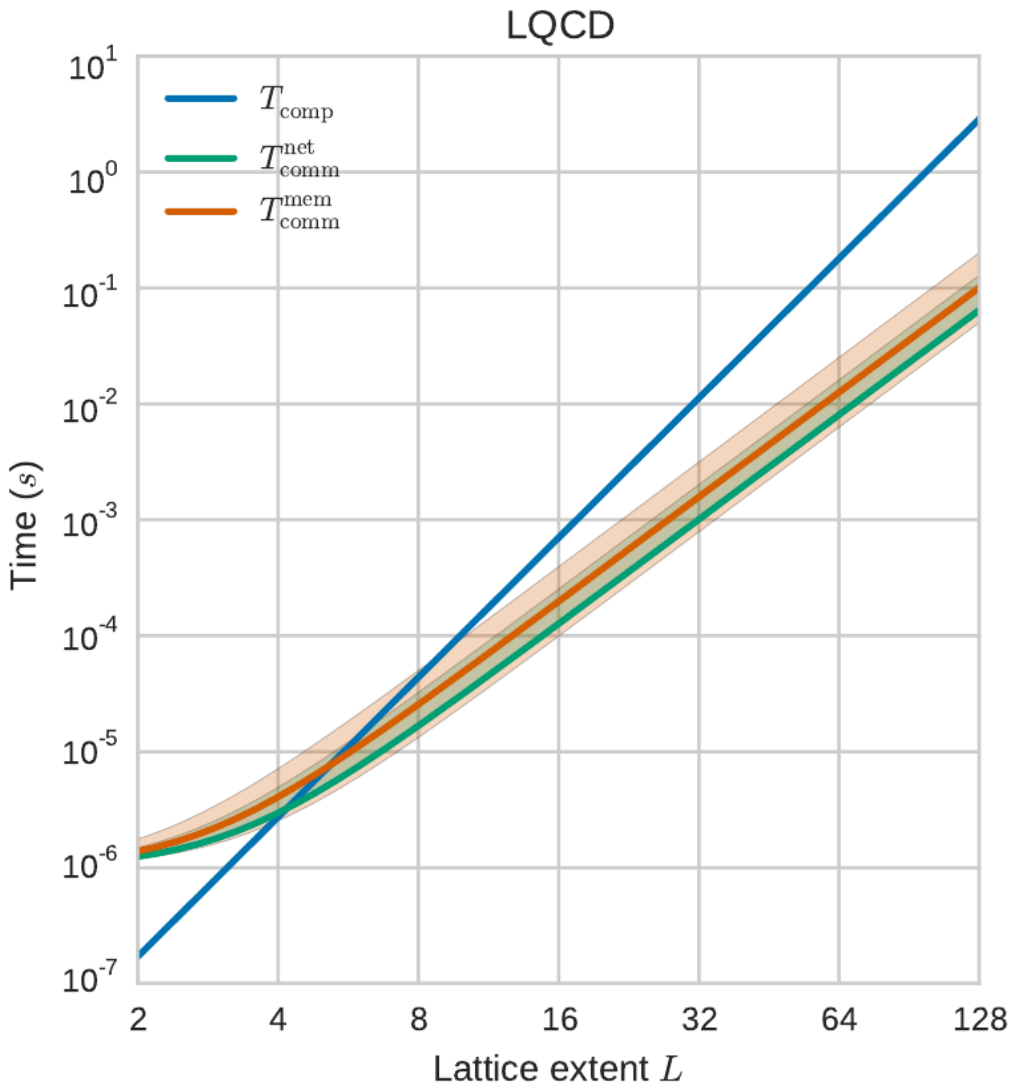
- $T_{mem} > T_{net}$
- $T_{comp} > T_{mem}$ for $L \geq 8$

Performance

- ~130 GFlop/s (SP) per AMC
- ~13 Gflop/s/W (SP) (AMC only)

Strong scaling limits

- Assume $V = 128^3 \times 256$ and $l^4 = 8$ per lane
  $\rightarrow$ 4096 AMC
- Upper limit of 0.5 Pflop/s
  - 7 PFlop/s at lower sustained



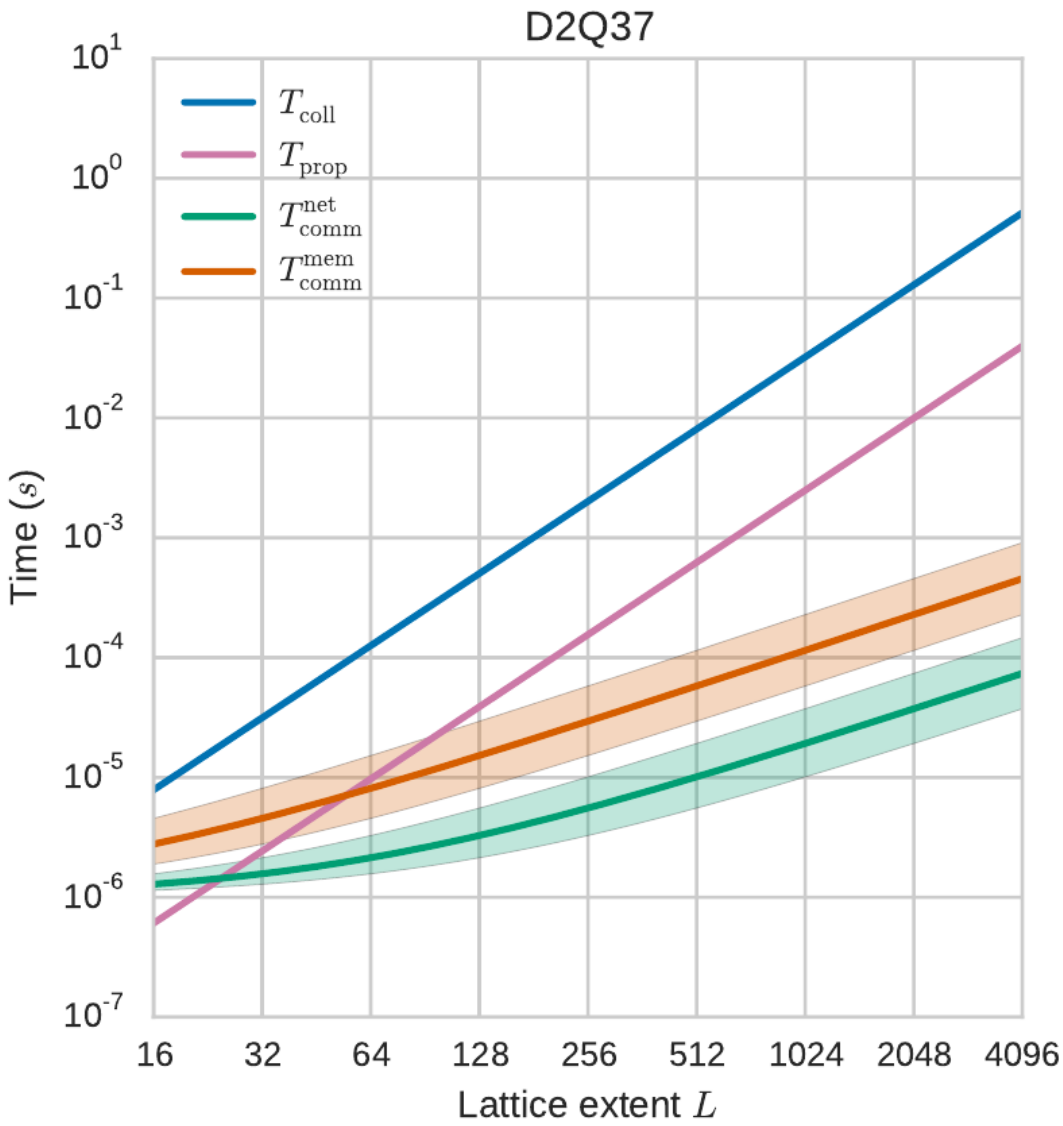
System level performance: D2Q37

Observations

- Time spent in collide kernel computations completely dominate

Performance

- ~175 GFlop/s (DP) per AMC
- ~17 GFlop/s/W (DP) (AMC only)



Architecture evaluation

Considered aspects

- Number of slices
- Number of scalar/vector registers, vectors register length
- Local Instruction Buffer size
- Load-store queue size
- Memory capacity and bandwidth

Conclusions LBM

- Collide performance limited by ISA
- Application could cope with longer vectors

Conclusions LQCD

- Performance limited by memory bandwidth
- Support for optimal SP complex arithmetics required

Summary and conclusions

Processing-in-memory architectures

- Continues to be an interesting architectural proposition
- Availability of products still unclear

Scientific computing applications on AMC

- Considered applications could exploit AMC efficiently
- Evaluation limited to relatively simple application kernels due to **not yet** available programming environment
- PIM attractive option for scientific computing

AMC architecture evaluation

- Architectural parameters matched application requirements well